Apple Computer Inc.
-A2L0008/030-0012 Printed in U.S.A. 10/78-2500

APPLE II
SERIAL INTERFACE CARD (A2L0008)
INSTALLATION AND OPERATING MANUAL

PLEASE READ THIS MANUAL BEFORE ATTEMPTING TO INSTALL THE SERIAL INTERFACE CARD INTO THE APPLE II.

SERIAL INTERFACE MANUAL
INSTALLATION AND OPERATING MANUAL

TABLE OF CONTENTS

SECTION	TITLE	PAGE
	INTRODUCTION	
I	INSTALLATION Installing the Serial Interface Compatibility with External Devices RS232 Connector Usage Current-Loop Operation with a Teletype	3
II	OPERATION Using the Serial Interface Preliminary Discussion of IN# and PR# Sending Output and Receiving Input	9
III	DEFAULT PARAMETERS AND THE DIP SWITCH Initialization Operating Parameters Setting the DIP Switch Defaults Permanent Defaults	15
IV	ACCESS TO OPERATING PARAMETERS Description of Serial Interface Operation Transmission Data Sequence Lower-Case Characters Changing Interface Parameters through Software Commands	19
V	DIRECT USE OF THE INTERFACE Transmitting a Character without Using PR#1 Batch Moves	27
VI	APPENDIX: SERIAL INTERFACE TIMING Table of Baud Rate Quantum Numbers Circuit Diagram Assembly Listing	31

APPLE II SERIAL INTERFACE CARD

INTRODUCTION

These are the fundamental abilities of the APPLE Serial Interface, using the nearly universal RS232 standard:

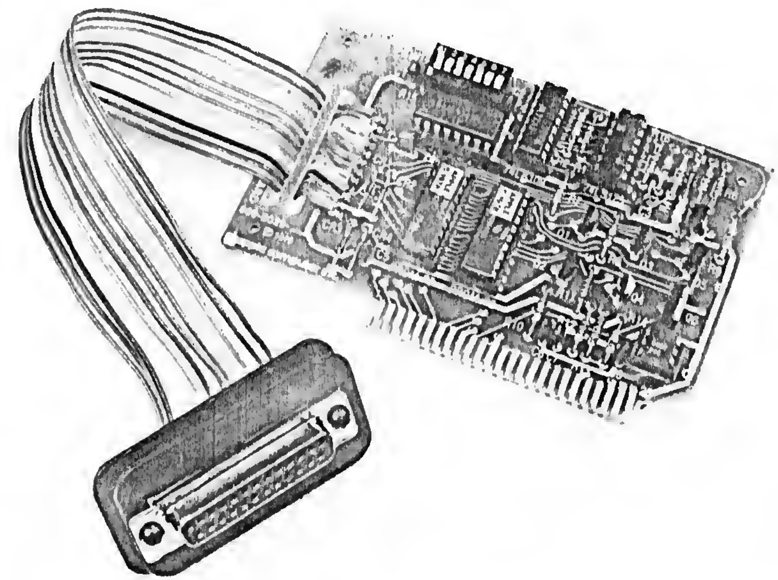
1. Output from the APPLE II can be sent to a serial printer or other external serial device, to the APPLE's TV screen, or to both. The Serial Interface can supply the necessary line-feeds with carriage-returns, etc.
2. Input for the APPLE II can be taken either from an external device or from the APPLE's keyboard, or from both simultaneously.
3. The APPLE II can handle half-duplex communications at rates from 75 to 19,2ØØ baud, in both directions, with a printer, another APPLE, a terminal, modem or other RS232 external device.
4. The Serial Interface can also be connected for current-loop operation with a Teletype.

While this document is intended primarily for APPLE users who are familiar with the RS232 interface, many of the terms and concepts will be explained.

I INSTALLATION

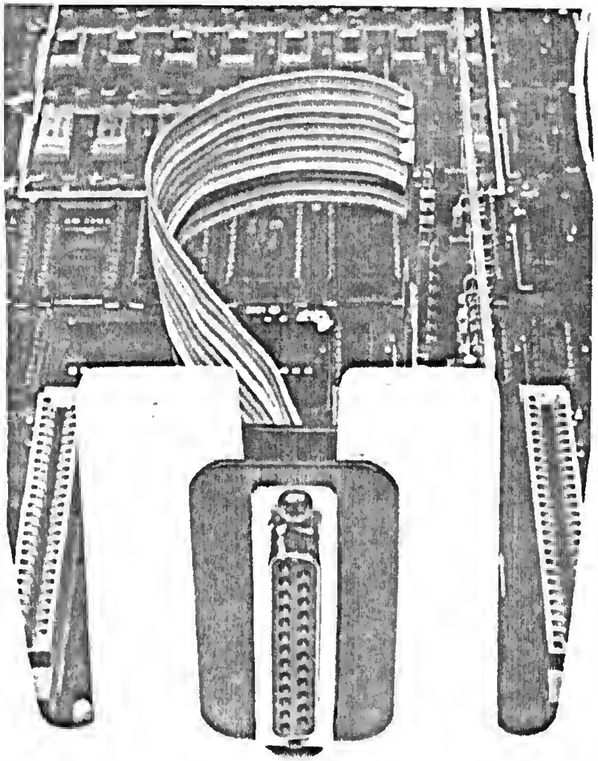
HOW TO INSTALL THE SERIAL INTERFACE

The Serial Interface consists of three parts: the Interface printed-circuit card itself, a female DB-25 connector, and a flat ribbon cable between them.



To install the Serial Interface, you will simply plug the Interface card into a socket inside the APPLE II, and then tighten a clamp to hold the DB-25 connector in place, as follows:

1. **Turn off the power switch** on the back of the APPLE II. This is important to prevent damage to the computer.
2. Remove the cover from the APPLE II. This is done by pulling up on the cover at the rear edge (the edge farthest from the keyboard) until the two corner fasteners pop apart. Do not continue to lift the rear edge, but slide the cover backward until it comes free.
3. Inside the APPLE II, across the rear of the circuit board, there is a row of eight long, narrow sockets called "**slots.**" The leftmost one (looking at the computer from the keyboard end) is slot #Ø, and the rightmost one is slot #7. Insert the "fingers" portion of the Serial Interface card into slot #1, the *second* socket from the left. The "fingers" portion will enter the socket with some friction and will then seat firmly. The Interface card may be placed in any slot except slot #Ø, the leftmost. However, APPLE's stan-



dard location for printer interfaces is slot #1 (the second from the left). This manual and most APPLE software for the Serial Interface are written assuming you have installed the Serial Interface card in slot #1.

4. Slip the DB-25 connector and its two metal plates as far down as possible into one of the three long vertical openings in the back of the APPLE II case. One plate goes on the inside of the case; the other plate goes on the outside of the case with the connector's flange on the outside of this plate. Any of the three large vertical openings may be used, but it is customary to use the middle one. Notice that the connector is not symmetrical. When seen from the back of the APPLE II, the longer side of the connector should be on the left (although it will work in either position).
5. Tighten the screws on the DB-25 connector just until the connector assembly can no longer be moved in the opening. Excessive tightening will cause the metal plates to bend.
6. Replace the cover of the APPLE II, remembering to start by sliding the front edge of the cover into position. Press down on the two rear corners until they pop into place.
7. The Serial Interface is installed, and the APPLE II may now be turned on.

COMPATIBILITY WITH EXTERNAL DEVICES

For communications between computers and computer-related equipment, the most widespread and universal standard is the RS232 standard. The RS232 standard specifies the electrical parameters, the form of the signals, and even the type of connector to be used in an interface. The APPLE Serial Interface complies with this standard.

The RS232 standard allows for a number of different communication speeds. These speeds are measured in terms of a unit called the "**baud**." Each multiple of 1Ø baud is equal to about 1 character sent or received per second; 3ØØ baud is roughly equal to 3Ø characters per second. The Serial Interface can operate at any of 256 different speeds, from 75 baud to 19,2ØØ baud.

Computers and their related devices do not actually send the keyboard characters themselves, of course. Each character is encoded in the form of electrical signals, and it is these electrical signals which are sent and received.

The APPLE Serial Interface can communicate with any device that specifies RS232 operation between 75 and 19,2ØØ baud. Many devices can operate at a number of speeds. Very often a set of switches or a rotary dial selects the baud rate. These external devices should be set to a particular baud rate before being connected to the APPLE. The highest baud rate available is usually preferred. The Serial Interface should be set for the same baud rate, using the first 3 levers of the Serial Interface's **DIP** switch (this is explained in the section, SERIAL INTERFACE OPERATING PARAMETERS). All common baud rates are listed in the section, SERIAL INTERFACE TIMING.

While such operation does not conform to the RS232 standard, the APPLE Serial Interface can also be operated in the current-loop mode necessary to communicate with a serial teleprinter such as the Teletype Model 33ASR.

RS232 CONNECTOR USAGE

The standard DB-25 connector, which is supplied with the Serial Interface, has 25 pins. Six of these are connected internally to the APPLE Serial Interface, but for most applications only three of them need be used. If you don't have a ready-made cable that can go from the Interface's DB-25 connector to the external device, then you will have to wire an interconnecting cable. A cable is just a number of electrically distinct wires that physically run alongside each other. When you wire the cable, you will have to refer to the DB-25 connector's pin numbers. These numbers are molded into the connector, although sometimes they are almost vanishingly small.

The following list describes the functions of the active pins on the Serial Interface connector. The other pins may be left unconnected.

PINS 4 & 5 These pins have been wired together (jumpered) at the Serial Interface card. No connection need be made to these pins.

PINS 6, 8 & 20 These pins have been wired together (jumpered) at the Serial Interface card. No connection need be made to these pins.

PIN 7 This is called "signal ground." It should be wired to pin 7 at the other end of the cable. If there is no connector at the other end of the cable, then the Serial Interface's pin 7 should be wired to a signal ground connection on the external device. (If this is insufficient information, any additional data would have to be supplied by the manufacturer or designer of the external device.)

PIN 2 The characters from the external device arrive at the computer via this pin.

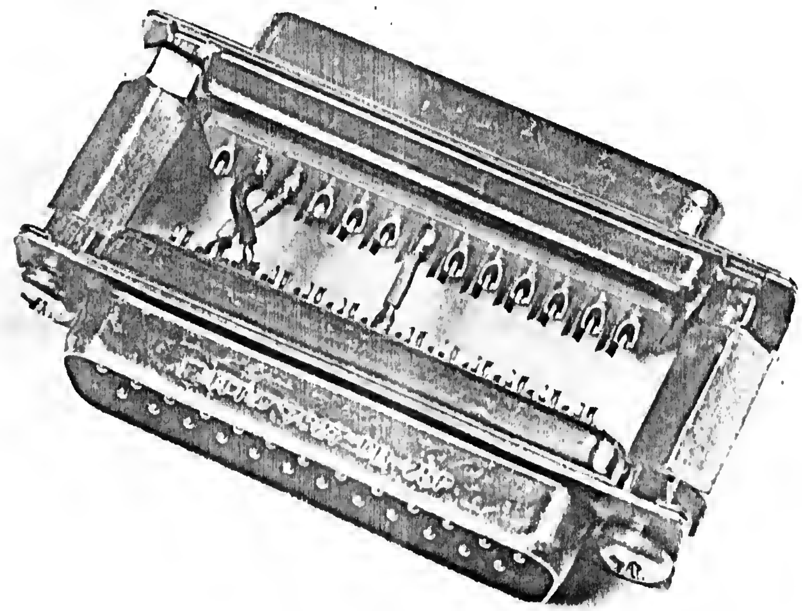
PIN 3 The characters leaving the computer, on their way to the external device, exit via this pin.

Pins 2 and 3 have been left for last since, if the external device end of the cable is another 25 pin connector, there are two ways that they might be wired. No damage is caused by wiring these pins the wrong way, but characters will not be sent out or received.

A. If the external device is a terminal or printer with an RS232 interface itself, then pin 2 on the APPLE's end of the cable should be wired to pin 2 at the external device's end of the cable. Similarly, pin 3 on the APPLE's end of the cable should be wired to pin 3 at the external device's end of the cable. Most of these devices, like the APPLE Serial Interface, also have a **female** DB-25 connector. Therefore your cable will (most likely) need to have a **male** DB-25 connector at *each* end.

B. If the external device is a modem, or another computer with a standard serial interface, then *its* interface will send characters out via pin 3 and receive characters via pin 2 just as the APPLE Serial Interface does. Therefore, you must wire pin 2 at the APPLE's end of the cable to pin 3 at the modem end of the cable; and wire pin 3 at the APPLE's end of the cable to

pin 2 at the modem end of the cable. Modems usually have a **male** DB-25 connector. Therefore, you will probably need a cable with a **female** DB-25 connector at the modem end, and a **male** DB-25 connector at the APPLE end.



Note that only the wires to pins 2 and 3 are involved. No other wires need be changed, no matter what external RS232 device is connected to the APPLE Serial Interface.

Most commercially prepared cables are simply extension cables: they connect identically numbered pins at the two ends of the cable. For use with a modem, you may have to re-wire the cable, interchanging the wires to pins 2 and 3 at one end. If you would rather not tamper with a ready-made cable, you can make an adapter with a male DB-25 connector at one end and a female DB-25 connector at the other end. It should be wired with pins 7 connected, and pins 2 and 3 cross-connected. This adapter, when connected between the APPLE's Serial Interface and a standard RS232 cable, allows the Interface to "talk" to most modems.

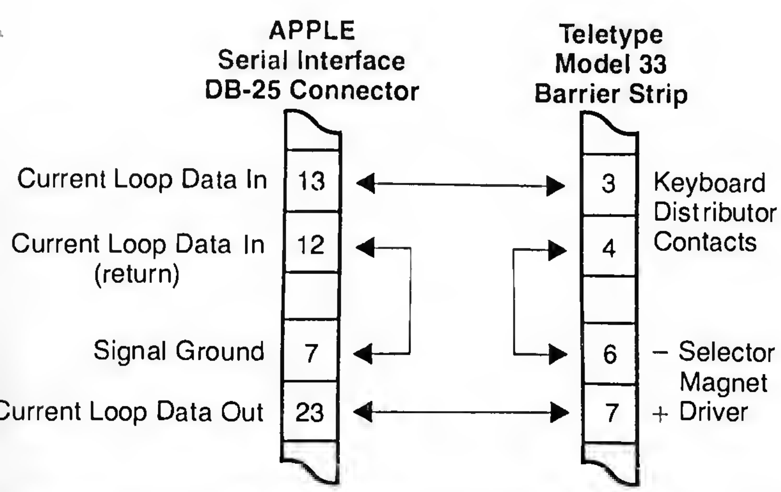
CURRENT-LOOP OPERATION WITH A TELETYPE

If you wish to use the APPLE Serial Interface to communicate with a current-loop teleprinter such as the Teletype Model 33ASR, you will be interested in three other Interface pins.

PIN 13 In current-loop operation, the characters from the Teletype arrive at the APPLE via this pin. This pin should be wired to terminal 3 on the Teletype Model 33's barrier strip.

PIN 12 The return path of the input current loop (the loop for characters arriving from the Teletype) should be connected to this pin. (In fact, the Serial Interface does not care which way current flows through this input loop. We have arbitrarily chosen pin 13 as the input and pin 12 as the return, but the roles of these two pins can be interchanged.) We suggest using signal ground (at pin 7) for the return path, in which case you should connect pin 12 to pin 7.

PIN 23 In current-loop operation, the characters leaving the APPLE, on their way to the Teletype, exit via this pin. Wire this pin to terminal 7 on the Teletype Model 33's barrier strip. The return path for this output current loop is also the signal ground at pin 7. For half-duplex operation, connect terminals 4 and 6 on the Teletype Model 33's barrier strip.



Caution! Pins 1 and 2 on the Model 33 barrier strip are connected to 12Ø Volts AC.

II OPERATION

USING THE APPLE SERIAL INTERFACE

The Serial Interface allows the APPLE II to communicate with other electronic devices which are *external* to the computer. These devices may be —to give a few examples—terminals, printers, or other computers. The Serial Interface can be controlled through BASIC programs or through assembly-language programs. It can also be controlled directly, by typing a few characters on the APPLE's keyboard.

In the following discussion, it will be assumed that you are familiar with the APPLE II BASIC Programming Manual, and that your APPLE II is operating in BASIC, with the Serial Interface installed in slot #1.

Here is a list of the most common tasks the Serial Interface is called upon to do, and the commands that accomplish them.

1. Send subsequent output to the Serial Interface.
   PR#1
2. Cancel the effect of PR#1, sending output only to APPLE's TV screen.
   PR#Ø
3. Accept subsequent input from the Serial Interface, as well as from the APPLE's keyboard.
   IN#1
4. Force the APPLE to convert all lower-case characters to upper-case, as they arrive from the external device.
   ESC U (type the ESC key, then type U)
5. Allow the APPLE to accept lower-case characters that arrive from the external device. If displayed on APPLE's TV screen, the lower-case characters will appear as upper-case characters in inverse video.
   ESC L

These tasks are more fully explained, and some fine points considered, in the next few pages.

To understand exactly how the Serial Interface operates, it is useful to think of the APPLE II as divided into three parts:

1. The APPLE's keyboard, which **generates** characters (when you type on it).
2. The APPLE's TV screen, which can **absorb** characters (and make them visible).
3. The APPLE's processor, or "brain," which can **control** the flow of characters, and act upon them.

You can also think of any external device as being able to **generate** characters or **absorb** characters, or both. The external device may or may not have a "brain," but this is not important in understanding the operation of the Serial Interface.

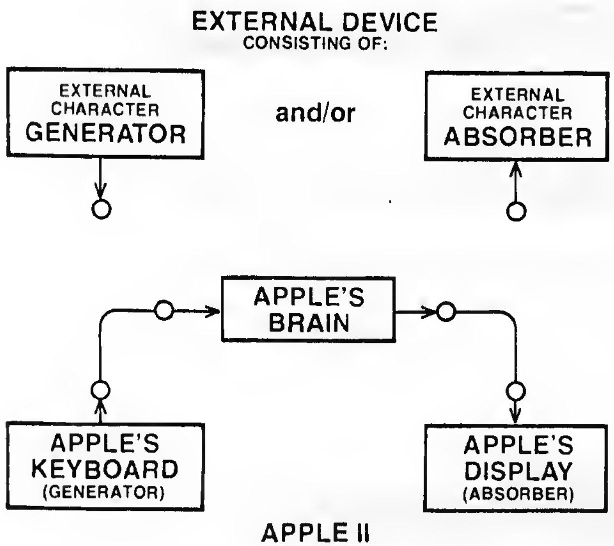


FIGURE 1

Figure 1 shows the three parts of the APPLE II, and the only parts of any external device that affect the Serial Interface. It also shows the normal interconnection between these parts. When the APPLE II is first turned on, it will ignore the external world, listening only to its own keyboard and displaying characters only on its own TV screen.

PRELIMINARY DISCUSSION OF THE IN# AND PR# COMMANDS

There are eight sockets, called "slots," on the back of the main circuit board inside the APPLE II. The leftmost one (as viewed from the keyboard end of the computer) is slot #Ø, and the rightmost one is slot #7. (See the section, HOW TO INSTALL THE SERIAL INTERFACE.) APPLE BASIC has two commands for selecting among these slots for input and output. In effect, when you first invoke BASIC, the commands
IN#Ø
and
PR#Ø
are automatically executed. The first of these commands, **IN#Ø**, tells the APPLE to
Take **IN**put from the APPLE keyboard.
And the second command, **PR#Ø**, instructs the APPLE to

Send **PR**inting to the APPLE's TV screen.
This is the "normal," or APPLE-alone condition shown in Figure 1. Now, however, if the command (or program statement)
IN#1
is executed, the APPLE will henceforth take its input from whatever is plugged into slot #1. Similarly, if the command
PR#1
is executed, all output will be sent to whatever is plugged into slot #1. If there is nothing plugged into the specified slot, then the system may hang, or your program may be erased, or other strange behavior may result. Notice that slot #Ø is special, and refers to the APPLE itself.

SENDING YOUR OUTPUT TO AN EXTERNAL DEVICE AND RECEIVING INPUT FROM AN EXTERNAL DEVICE

In the following examples, the commands
PR#1
and
IN#1
will be typed on the APPLE's keyboard. If you have put your Serial Interface into slot #1 (the second one from the left, as described in the section, HOW TO INSTALL THE SERIAL INTERFACE) the commands will work exactly as shown. If you use some other slot, you will have to substitute the number of that slot. **Slot #Ø may not be used for the Serial Interface.**

Attach an appropriate cable (see the section, RS232 CONNECTOR USAGE) from the DB-25 connector of the Serial Interface to the external device with which you wish to communicate. Reset your APPLE II by pressing the RESET key, and invoke BASIC typing a CTRL B (CTRL B means depressing the B key while simultaneously holding down the key marked CTRL). If you are not familiar with this procedure, see your APPLE II BASIC Programming Manual. When the prompt character appears, type the command
PR#1
(and press the RETURN key, of course) . From now on, any characters you type will be sent out through the Serial Interface to the external device. The characters will appear on the APPLE's TV screen only if levers 5 and 6 of the Serial Interface's **DIP** switch (located directly on the Interface card itself, near its upper edge) are **ON** when **PR#1** is typed. (This is more fully explained in the section, SERIAL INTERFACE OPERATING PARAMETERS.) Characters coming in from the external device will be ignored. The operation of the system after you type **PR#1** is shown in Figure 2.

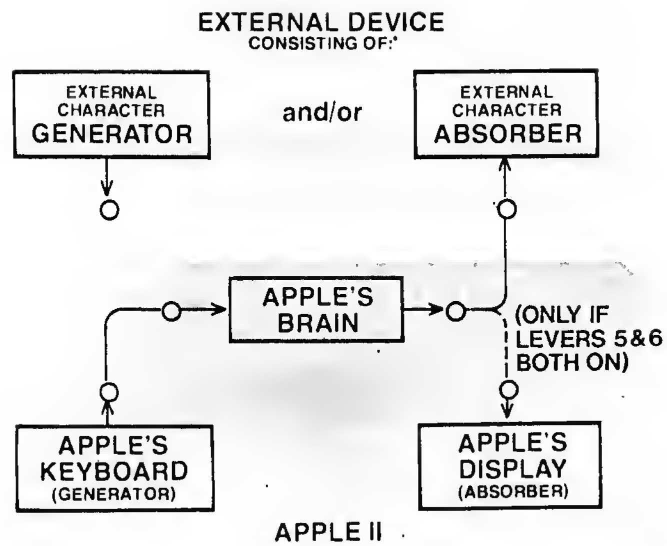


FIGURE 2

Operation of the system after executing PR#1

The APPLE's "brain" is still connected, and the command

PR#Ø

will restore normal (Figure 1) operation, in which the APPLE's output characters are not sent through the Serial Interface. Normal operation can also be restored by pressing the RESET key and then typing a CTRL C, but this option is not available if the Interface is being controlled by a program.

To let the external source of characters control the APPLE II, use the command

IN#1

After this command, the APPLE will accept input from the external device connected to the Serial Interface, as well as from its own keyboard. Figure 3 shows this condition. If there is no external device connected to the Serial Interface, the system will "hang" after this command. Use RESET CTRL C to recover.

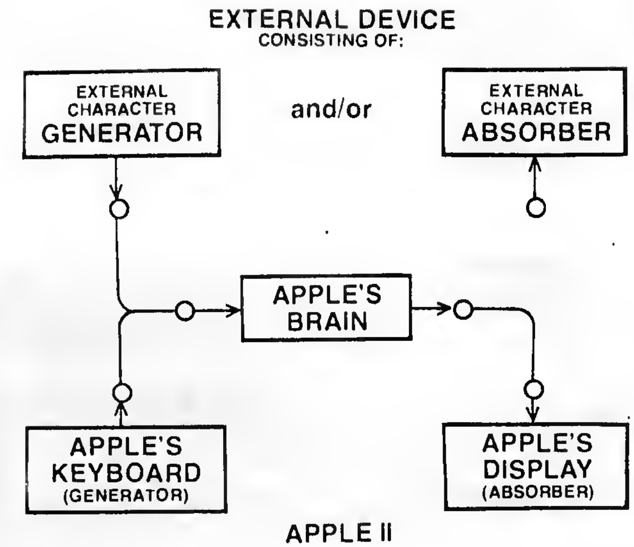


FIGURE 3

Operation of the system after executing IN#1

Normal operation is restored if the command

IN#Ø

is typed on the APPLE's keyboard. Normal operation is also restored if the external device sends the command

IN#Ø

Pressing the RESET key and typing a CTRL C on the APPLE's keyboard will also restore normal operation; but this cannot be done from a program, and will not be mentioned again.

Typing **both** commands,

PR#1

and

IN#1

will give the external device full control of the APPLE II. In this "remote mode" (shown in Figure 4), a friend could use your APPLE from across the country—or across the room.

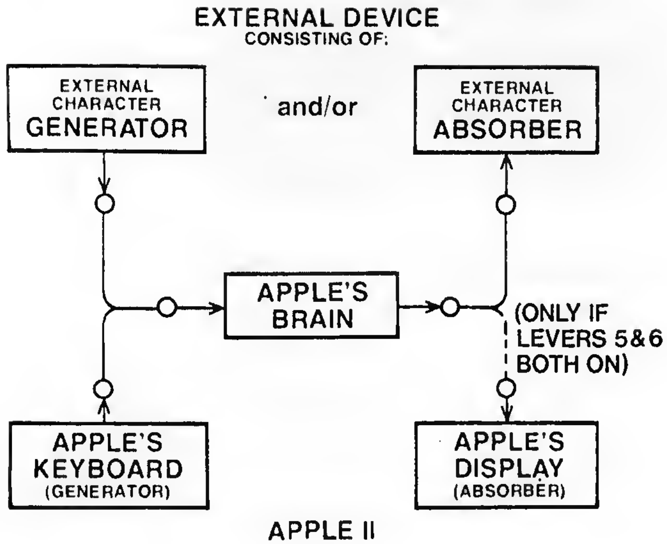


FIGURE 4

Remote Mode: operation of the system after executing PR#1 and IN#1.

III DEFAULT PARAMETERS AND THE DIP SWITCH

INITIALIZING THE SERIAL INTERFACE

Before the Serial Interface can be used, it must be **initialized**. Initializing the Interface sets all of the Interface operating parameters to their **default** values. Assuming slot #1, the Interface is initialized each time either of the following BASIC commands is typed:

PR#1 RETURN or **IN#1** RETURN

and each time any of the following Monitor commands are typed:

1 CTRL P RETURN or 1 CTRL K RETURN or **C1ØØG** RETURN

When used within a *program*, a command (such as PR#1) that transfers APPLE's output to the Serial Interface does *not* initialize the Interface until the first character is actually sent out (with a PRINT statement, for instance). Similarly, if during a *program* a command (such as IN#1) tells the APPLE to get its input from the Serial Interface, the Interface is *not* initialized until the APPLE actually looks for its first input character (in an INPUT statement, for instance).

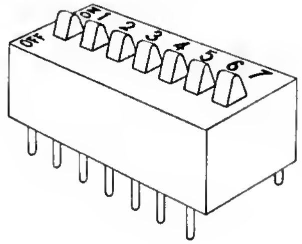
SERIAL INTERFACE OPERATING PARAMETERS

The Serial Interface has ten user-definable operating parameters. Each time the Serial Interface is initialized, the ten operating parameters are given their **default** values. Five of the default values are determined by the 7 levers of the Serial Interface's **DIP** switch (located on the Interface's printed-circuit card, near the upper edge). The **DIP** switch levers set the default values for these five operating parameters: **Baud Rate** (levers 1, 2 and 3), **Carriage Return Delay** (lever 4), **Line Width** plus **APPLE Video** (levers 5 and 6), and **Line Feed** (lever 7). Changing the settings of the **DIP** switch levers *after* initialization has no effect until the *next* initialization.

SETTING THE DIP SWITCH DEFAULTS

1) Baud Rate

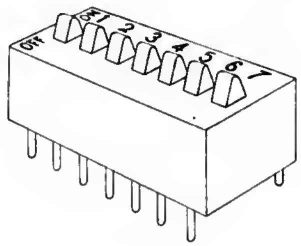
DIP Switch Levers 1	2	3		Default Baud Rate
On	On	On	=	110 baud
Off	On	On	=	134.5 baud
On	Off	On	=	300 baud
Off	Off	On	=	1200 baud
On	On	Off	=	2400 baud
Off	On	Off	=	4800 baud
On	Off	Off	=	9600 baud
Off	Off	Off	=	19200 baud



On initialization, the settings of **DIP** switch levers 1, 2 and 3 determine the rate at which bits may be transmitted to the external device. 300 baud is 300 bits per second. Under default conditions, each character is transmitted using 11 bits (1 start bit, 8 data bits, and 2 stop bits).

2) Carriage Return Delay

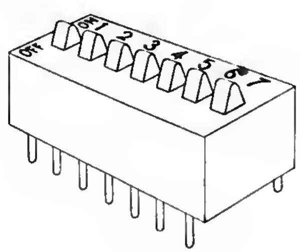
DIP Switch Lever 4		Default Car. Return Delay
On	=	Disabled
Off	=	Enabled



If **DIP** switch lever 4 is Off (Delay Enabled), the Serial Interface will wait briefly (approximately ¼ second) after transmitting a carriage return, to allow the printer to complete this movement. If you are transmitting to an external TV screen, this delay is probably unnecessary, and lever 4 may be turned On (Delay Disabled).

3) Line Width plus APPLE Video

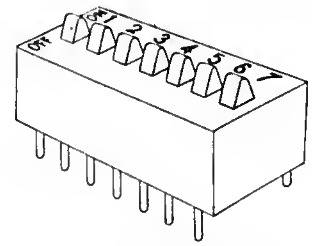
DIP Switch Levers 5	6		Default Line Width	Default APPLE Video
On	On	=	40 Char/Line	Enabled
Off	On	=	72 Char/Line	Disabled
On	Off	=	80 Char/Line	Disabled
Off	Off	=	132 Char/Line	Disabled



After a carriage return, the settings of **DIP** switch levers 5 and 6 determine the maximum number of characters transmitted before the Serial Interface will force another carriage return to be sent out. Characters will be displayed on the APPLE's TV screen only if the default line width is set to 40 characters per line (levers 5 and 6 On). After initialization, the line width can be changed from 40 characters per line, but the display on the APPLE's TV screen will not correspond to the display on the external device, as transmitted carriage returns are not accompanied on the APPLE's TV screen by line feeds.

4) Line Feed

DIP Switch Lever 7		Default Line Feed
On	=	Disabled
Off	=	Enabled



If **DIP** switch lever seven is Off (Line Feed Enabled), the Serial Interface will transmit a line feed after each carriage return it transmits. If the external device automatically supplies its own line feed after each carriage return received, you can set lever seven to On (Line Feed Disabled) to avoid double-spacing.

PERMANENT DEFAULTS

During each initialization, the five remaining operating parameters are set to their **permanent** default values:

1. **Parity** defaults to its disabled condition (no parity bit).
2. **Checksum** defaults to its disabled condition (no checksum character).
3. **Lower-Case** defaults to its disabled condition (converts all incoming lower-case characters to upper-case).
4. **Number of Data Bits** defaults to 9 (8 data bits plus one start bit).
5. **Number of Stop Bits** defaults to 2.

These parameters will be explained in the following section; they can only be changed by software commands *after* initialization.

IV ACCESS TO OPERATION PARAMETERS

DESCRIPTION OF SERIAL INTERFACE OPERATION

For most applications, the default operating parameters (both those that are fixed and those that can be set with the **DIP** switch) will be just what you need: your parameters will be set each time the Interface is initialized. In that case, the following section will be interesting but not necessary. However, the Serial Interface is designed to be very flexible, so that its operating parameters can be easily modified for use in a wide variety of special applications. This section gives a rather detailed description of the Serial Interface's operation. The following section shows you how to make any of the many possible modifications to that operation, should they be necessary.

Each character that is sent out through the Serial Interface is transmitted as a series of bits, in the following sequence:

1. One "Start Bit," a "low" voltage which tells the external device that a character is going to be transmitted.
2. From two to eight "Data Bits" (default is eight bits; can be changed by the user), a sequence of "high" and "low" voltages that represent the actual character code being transmitted. The default is eight bits because the APPLE normally handles data in eight-bit groups. If your external device sends and receives data in groups of fewer than eight bits, you must set the Serial Interface to send and receive these smaller groups.
3. *If* enabled, one "Parity Bit" (default is no parity bit; even or odd parity can be enabled by the user). This is a transmission-accuracy checking bit which the external device looks at for errors and then discards.

   The parity bit is found as follows: all the 1-bits in the actual character code are added together, and the result's evenness or oddness is compared with the type of parity-checking selected. For instance, the binary ASCII code for the letter S is 1Ø1ØØ11; the sum of the 1-bits is 4, an *even* number. If *Even* parity has been enabled, the comparison is *true*, and a Ø-bit is sent at the end of the character. If *Odd* parity has been enabled, the comparison is *false*, and a 1-bit is sent. If parity has not been enabled, no extra bit is sent. Check your device's operation manual to see if it sends and receives parity bits.
4. From one to 127 "Stop Bits" (default is two bits; can be changed by the user), a "high" voltage which tells the external device that a character has been completely transmitted. Each external device requires a particular number of stop bits after every character; see the device's operation manual.

This same sequence will be used by the external device when it transmits to the APPLE. Timing is very important to correct transmission and reception. The Serial Interface sends out and receives bits at fixed intervals of time set by the "Baud Rate" (default is set by **DIP** switch levers 1, 2 and 3; can be

changed by the user). The Serial Interface and the external device *must* be set to the same baud rate and parity option, in order to interpret the sequence of "high" and "low" voltages correctly. The same is true when the external device is transmitting characters to the APPLE Serial Interface.

When the Serial Interface has sent out the number of characters set by the "Line Width" (default is set by **DIP** levers 5 and 6; can be changed by the user), it transmits a "Carriage Return" to the external device. After sending a carriage return, it may wait during a fixed ¼-second "Carriage Return Delay" (if enabled: default set by DIP lever 4; can be changed by the user) before sending the next character, to allow the printer to complete this movement. Then the Interface may send a "Line Feed" (if enabled: default set by **DIP** lever 7; can be changed by the user), so that subsequent characters will appear on the following line.

Finally, each time it completes sending 256 characters in a "Batch Move," the Serial Interface may send a "Checksum" character (default is no checksum character; can be changed by the user). This is a transmission-accuracy checking character which the external device looks at for errors and then discards. The checksum character is found by XORing the previous 256 characters, as follows: the second character is XORed with the first, the third with that result, the fourth with *that* result, and so on. Check your device's operation manual to see if it sends and receives checksum characters. During Batch Moves, the Serial Interface and the external device must be set to the same checksum option.

LOWER-CASE CHARACTERS

While the APPLE generates and displays characters only in upper case (capital letters), your external device may generate both upper-case and lower-case characters. When lower-case characters are received by the APPLE through the Serial Interface, they can be treated in two different ways:

1. Convert all incoming lower-case characters to upper-case characters (incoming upper-case characters are not affected). Since this is the usual APPLE mode, any incoming characters displayed on the APPLE's TV screen will look fine. This is the permanent default condition; but it can be changed by the user, after initialization.
2. Accept all incoming lower-case characters as lower-case (again, incoming upper-case characters are not affected). APPLE's TV screen *display* of characters is designed for upper-case only, so any display of incoming lower-case characters will look strange. If the characters are being displayed as they arrive from the external device, APPLE will show the lower-case characters as upper-case characters in *inverse video* (black letters on a white background). Once stored in APPLE's memory, lower-case characters will be displayed (when LISTed, for instance) as a strange assortment of upper-case characters in normal (white on black) video. There is one exception to this: if the stored lower-case characters are being displayed as they are sent out through the Serial Interface, they will again appear as upper-case characters in inverse video. Note that these peculiar *displays* do not reflect the lower-case characters themselves: in this mode they are *stored* correctly in APPLE's memory, and may be printed correctly on any appropriate external device. Note also that this mode does not add any capability to *generate* lower-case characters from the APPLE keyboard.

CHANGING SERIAL INTERFACE PARAMETERS THROUGH SOFTWARE COMMANDS

Ten of the Serial Interface parameters can be changed from their initialized (default) values, through the use of commands in machine language or BASIC. Once an Interface parameter is set by a software command, that parameter remains unchanged until the Serial Interface is reinitialized or the parameter is reset by another software command. For more discussion of the various parameters functions, see the previous section.

In the following descriptions, the letter "s" refers to the number of the printed-circuit board slot inside the APPLE, in which the Serial Interface card is installed (see the section, HOW TO INSTALL THE SERIAL INTERFACE).

1. **BAUD RATE** (assembly-listing variable: BRATE)

   Memory location 1144+s ($478+s, in Hexadecimal) contains APPLE's baud "quantum" number, which specifies how many "quanta" the APPLE is to wait between sending out bits through the Serial Interface. One quantum equals 53 APPLE II cycles (51.94 microseconds) per transmitted bit. The default value is set with levers 1, 2 and 3 of the Interface card's **DIP** switch (see the section, SETTING THE DIP SWITCH DEFAULTS). From BASIC, to change the baud rate from the default value to B use the command
   POKE 1144+s, r
   where r is the integer, from Ø through 255, that is closest to
   1/(.ØØØØ5194 * B)
   For further information, see the section SERIAL CARD TIMING.

2. **STOP BITS** (assembly-listing variable: STBITS)

   Memory location 1272+s ($4F8+s, in hexadecimal) contains the Number of Stop Bits (Note: the one parity bit is *included* in this number, if

parity is enabled). The default value is 2 stop bits (and no parity bit). To change the number of Stop Bits from BASIC, use the command
POKE 1272+s, r
where r is an integer, from1 through 127. To determine the correct number of stop bits for your external device, see the external device's operation manual.

Note: you must add the one parity bit to the Number of Stop Bits, if parity is enabled.

3. **PARITY/CHECKSUM OPTIONS** (assembly-listing variable: STATUS)

Memory location 14ØØ+s ($578+s, in hexadecimal) contains a number, the lower three bits of which determine two parity options (enable/disable and even/odd) and one checksum option (enable/disable). If the remote device with which your Serial Interface is communicating requires a parity bit to be sent or received with each character, you can tell your Serial Interface to do this task. You can also specify which type of parity check (even parity or odd parity) is to be sent and received. If your remote device requires that a checksum be sent after every 256 characters in a Batch move, you can tell the Serial Interface to send one. To decide whether your external device requires either a parity bit or a checksum character (or both), consult the device's operation manual. The three Parity/Checksum options are changeable from BASIC by using the command
POKE 14ØØ+s, r
where r is an integer from Ø through 7. The actual value that r should be assigned is determined as follows:

Bit Ø:	1 = odd parity Ø = even parity	(This is the least significant, or rightmost, bit.)
Bit 1:	1 = no parity Ø = parity enabled	(initial default value)
Bit 2:	1 = no checksum Ø = checksum enable	(initial default value)

First determine whether or not a parity bit need be sent (Bit 1). If yes, then decide whether the parity should be odd or even (Bit Ø). Also, determine whether or not a checksum character need be sent during Batch moves (Bit 2). For example, let's assume that an even parity bit must be sent, with no checksum. Bit Ø gets a value of Ø, Bit 1 gets a value of Ø, and Bit 2 gets a value 1. This binary number 1ØØ is converted to its decimal equivalent of 4 and POKE'ed (assuming slot #1):
POKE 14Ø1, 4

4. **INPUT/OUTPUT BUFFER** (assembly-listing variable: BYTE)

Memory location 1656+s ($678+s, in hexadecimal) is the input buffer for the individual character that has just been received through the Serial Interface from the external device. Assuming the Interface is in slot #1, the BASIC command
PRINT PEEK (1657)
will print on the APPLE's TV screen the ASCII value of the character just received.

5. **LINE WIDTH** (assembly-listing variable: PWDTH)

Memory location 1784+s ($6F8+s, in hexadecimal) contains the "Printer Width," or number of characters per line. After transmitting this number of characters, the Serial Interface will then transmit its Carriage-Return sequence. To change the number of characters per line, from BASIC, use the command
POKE 1784+s, r
where r is an integer, from Ø through 255, specifying the number of characters per output line. To determine the maximum line width for your external device, consult the device's operation manual.

Note: if r is set to *zero*, the Serial Interface will not force *any* carriage returns to be transmitted. The output characters will be transmitted in a continuous stream.

6. **DATA BITS** (assembly-listing variable: NBITS)

Memory location 1912+s ($778+s, in hexadecimal) contains the number of Data Bits, plus one for the start bit. In the APPLE, data is handled in groups containing eight bits. If you are communicating with an external device which *also* handles data in eight-bit groups, the default Number of Data Bits is perfect (8 data bits plus 1 start bit). However, if your external device handles data in groups of *fewer* than eight bits, you must set the Serial Interface to send and receive these smaller data groups.

When *receiving* data groups of fewer than eight bits, the Serial Interface will supply 1's to fill the remaining high-order bits of each eight-bit group in APPLE's memory. Similarly, when the Serial Interface is *transmitting* data groups of fewer than eight bits, the unused high-order bits in each of the APPLE's eight-bit data groups must be set to 1's.

To change the Number of Data Bits from BASIC, use
POKE 1912+s, r
where r is an integer, from 3 (2 data bits plus one start bit) through 9 (8 data bits plus one start bit).

Note: to calculate r, you must add one start bit to the number of data bits. If r is set to less than the default value of 9 (8 data bits plus one start bit), you must set the unused high-order data bits to *ones* before transmitting the data. Received data will also have unused high-order data bits set to *ones*.

Example: Binary Coded Decimal is a code for sending numbers in four-bit data groups. The BCD code for the number 7 is 0111. If the Number of Data Bits, r, is set to 5 (4 data bits plus 1 start bit), BCD for the number 7 must be stored in the APPLE's eight-bit byte as 1111Ø111 before the data group Ø111 can be transmitted. Similarly, if the data group Ø111 is received by the Serial Interface, it will be stored in the APPLE's eight-bit byte as 1111Ø111.

7. OPERATION MODES (assembly-listing variable: FLAGS)

Memory location 2Ø4Ø+s ($7F8+s, in hexadecimal) contains a number, four of whose bits determine four separate modes of operation. To alter the operation modes from BASIC, use the command
POKE 2Ø4Ø+s, r
where the value of r is determined by use of the following table:

r's Binary Bit	(Decimal Equiv., If Bit=1)	Operation Set By Bit Value	Default
Bit#Ø	(1)	1 = Line feed after carriage return Ø = No line feed	(Set by Lever 7)
Bit#5	(32)	1 = Lower-case input enable Ø = Convert lower-case to Upper-case	(Permanent Default)
Bit#6	(64)	1 = No delay after carriage return Ø = Carriage return delay enable	(Set by Lever 7)
Bit#7	(128)	1 = No display on APPLE's TV Ø = APPLE's display enabled	(Set by Levers 5&6)*

*APPLE's TV display is only enabled during initialization if DIP switch levers 5 and 6 are both On.

For example, let us assume that you wish to have line feeds, upper-case only, carriage return delay, and no APPLE display. This would require that bits Ø and 7 have a value of 1, and bits 5 and 6 have a value of Ø. Add up the decimal equivalents of all of the bits that were assigned the value 1 (the decimal equivalents are the numbers in parentheses, next to

the Bit #'s). The decimal equivalents for bits Ø and 7 are 1 and 128 respectively; therefore the total of the decimal equivalents is 129. This value is assigned to r, and POKE'ed (assuming slot #1):
POKE 2Ø41, 129

If you wish to change only Bit #5 (lower-case input enable/convert), you can do so with the following commands:

ESC U (Press and release the ESC key, and then type U)

This changes r's Bit #5 to a zero, the default value. After this command, all lower-case characters arriving through the Serial Interface from an external device will be converted to upper-case characters. Incoming upper-case characters are not affected. This is the APPLE's usual mode, so any APPLE display will look fine.

ESC L

This changes r's Bit#5 to a one. After this command, lower-case characters arriving through the Serial Interface from an external device will be stored as lower-case characters in APPLE's memory. Upper-case characters are not affected. Since the APPLE was designed for upper-case characters only (BASIC will accept lower-case characters only in quoted strings), any APPLE *display* of these lower-case characters will look strange on the TV screen. See the previous section for details. However, the characters are *stored* correctly, and may be printed correctly on any appropriate external device.

Note: the commands ESC U and ESC L are Serial Interface *input* commands. They will have no effect unless the Interface has been initialized for input (by IN#1, for instance).

8. TAB

The TAB and comma functions in Integer BASIC (HTAB in APPLESOFT) will sometimes work in conjunction with the Serial Interface, but have several restrictions (fewest for comma-tabbing). A TAB of less than 18, if it would end directly on a character already printed, may be simply tabbed from that character's position. No TAB can cause printing to occur to the *left* of the last printed character on the current line. An attempt to do so usually causes printing to occur in the first available position to the *right* of the last printed character. Both Integer BASIC's TAB and APPLESOFT's HTAB send out a carriage return for every 4Ø positions in the tab instruction, and then tab the remaining positions. For tabbing to any position (including those beyond position 4Ø), you can use the BASIC command
POKE 36, r
where r is an integer, from Ø through 255, equal to the number of print

positions to be tabbed. This command suffers most of TAB's restrictions, except for the 4Ø-position limit. In APPLESOFT, the TAB function (used inside a PRINT statement) can also cause tabbing of more than 4Ø positions.

V DIRECT USE OF THE INTERFACE

TRANSMITTING A CHARACTER WITHOUT USING PR#1

Occasionally, it is useful to send a character out through the Interface *without* using a PR# command to change the "output vector" (the system pointer that tells your APPLE where to send its output, normally to its TV screen). To use the Serial Interface *directly*, follow these two steps:

1. Into APPLE's accumulator, put the ASCII code of the character to be sent.
2. CALL −16384 + (256 * s)

where s is the Interface's slot number (the equivalent hexadecimal location to CALL is $CsØØ). There are various ways to get a number into the APPLE's accumulator, but one way is to write a very small machine-language subroutine to do it, and then CALL that subroutine from your BASIC program. To begin, press the RESET key to enter the Monitor (prompt character: *), and then (assuming slot #1) type

3ØØ: A9 11 4C ØØ C1 RETURN

Check your work by typing
3ØØL

Ignoring most of the resulting display, the first two lines should look like this:

```
Ø3ØØ-   A9 11      LDA   #$11
Ø3Ø2-   4C ØØ C1   JMP   $C1ØØ
```

The first instruction (at hexadecimal location $3ØØ) tells the APPLE to **L**oa**D** the **A**ccumulator with the number in the next location (hexadecimal location $3Ø1). For now, that number is $11. The second instruction is equivalent to CALL −16128 in BASIC: it tells the APPLE to **J**u**MP** to hexadecimal location $C1ØØ, which starts the Serial Interface character output routine. To use this subroutine in your BASIC program, you must first put into hexadecimal location $3Ø1 (that's location 769, in decimal) the ASCII code for the character you want the Serial Interface to send out. Then you will CALL the subroutine at hexadecimal location $3ØØ (768, in decimal). Here is a short program that uses the above machine-language routine to send out one character at a time:

```
1Ø INPUT "LETTER?", L$
2Ø POKE 769, ASC(L$)
3Ø CALL 768
4Ø GOTO 1Ø
```

BATCH MOVES

At times it is useful to send or receive large amounts of information very quickly. This can be accomplished through use of a *Batch Move*. The Batch subroutines are "utility" routines. They are intended to be used for special

applications such as Data Collection, Mass Storage and Retrieval Systems, and sending program sequences to control external devices. To understand how to use the Batch routines on the simplest level, refer to the examples below. For an example of using the Batch Moves from BASIC, see the next section, BATCH MOVES FROM BASIC.

Note: before using the Batch routines, you must have initialized the Serial Interface (by typing PR#1, for instance) and you must have set the desired parity and checksum parameters. The Batch Move commands deal directly with the Serial Interface (not through the input and output "vectors" set by IN#and PR#); therefore these commands are the same, regardless of which slot contains the Interface card.

1. Batch Output

When in the Monitor (prompt character: *), type

3F8: 4C 41 C9 RETURN

This prepares the APPLE to jump to hexadecimal location $C941 in the Serial Interface's Read-Only Memory when a CTRL Y is typed on the keyboard. This jump causes the Batch Output routine to execute. When you are ready to actually send the data, type

addr1 . addr2 CTRL Y RETURN

where "addrl" is the hexadecimal starting address of the data, and "addr2" is the hexadecimal ending address of the data. For example, if we wanted to send the information that is stored in memory from address $2ØØØ through address $3FFF, we would type

2ØØØ.3FFF CTRL Y RETURN

As soon as the return is typed, the data from address 2ØØØ to address 3FFF will be sent through the Serial Interface from the APPLE to the external receiving device.

2. Batch Input

When in the Monitor, type

3F8: 4C 3D C9 RETURN

This prepares the APPLE to jump to hexadecimal location $C93D in the Serial Interface's Read-Only Memory when a control Y is typed on the keyboard. This jump causes the Batch Input routine to execute. When you are ready to actually receive the data, type

addr1 . addr2 CTRL Y RETURN

where "addrl" is the hexadecimal starting address in which the incoming data will be stored, and "addr2" is the hexadecimal ending address in which the incoming data will be stored. For example, if we wanted to receive data from an external device, and store it in our APPLE's memory from address $4ØØØ through address $5FFF, we would type

4ØØØ.5FFF CTRL Y RETURN

As soon as return is typed, the serial data can be sent by the external transmitting device to the Serial Interface. As it is received, the incoming data will be stored in your APPLE's memory from address 4ØØØ through address 5FFF.

Note: when the Serial Interface is instructed to receive a batch move, the cursor on the receiving APPLE's TV screen disappears, and the Interface waits patiently until *all* the specified locations have been filled with received data. Then the cursor returns.

BATCH MOVES FROM BASIC

While it is easiest to use the Batch routines from the Monitor, it is also possible to do Batch Moves from BASIC. In the following discussion these definitions will hold:

BAL = Beginning Address Low
(the two rightmost digits of the 4-digit hexadecimal starting address for the move, converted to decimal)

BAH = Beginning Address High
(the two leftmost digits of the 4-digit hexadecimal starting address for the move, converted to decimal)

EAL = Ending Address Low
(the two rightmost digits of the 4-digit hexadecimal ending address for the move, converted to decimal)

EAH = Ending Address High
(the two leftmost digits of the 4-digit hexadecimal ending address for the move, converted to decimal)

Suppose you wish to send someone a picture from your APPLE's high-resolution screen (page 1). The memory for this screen lies between 8K and 16K, from hexadecimal address $2ØØØ to hexadecimal address $4ØØØ. For the Beginning Address $2ØØØ:

BAL = $ØØ (hex) = Ø (decimal)
BAH = $2Ø (hex) = 32 (decimal)

For the Ending Address $4ØØØ:

EAL = $ØØ (hex) = Ø (decimal)
EAH = $4Ø (hex) = 64 (decimal)

1. Batch Output from BASIC

The following BASIC program does the same task that the Monitor Batch Output command did. See the discussion of the Monitor Batch Output for more details.

```
1Ø PR#1 : PRINT " "                  (initializes Interface)
2Ø POKE 6Ø, BAL : POKE 61, BAH (sets starting address)
3Ø POKE 62, EAL : POKE 63, EAH (sets ending address)
4Ø CALL -14Ø15                       (jumps to Output Routine at $C941)
5Ø PR#Ø                              (returns to normal TV output)
6Ø END
```

2. Batch Input from BASIC

In the Batch Output program, above, change line 4Ø to

```
4Ø CALL -14Ø19                       (jumps to Input Routine at $C93D)
```

The resulting BASIC program does the same task that the Monitor Batch Input command did. See the Monitor Batch Input discussion for more details. Note that the IN#1 is *not* necessary for accepting input through the Serial Interface, because CALL -14Ø19 deals directly with the Interface (not through the input and output "vectors" set by PR#and IN#). In fact the PR#1 in line 1Ø was necessary *only* to initialize the Interface;

1Ø CALL -16128

would have done as well.

VI APPENDIX: SERIAL INTERFACE TIMING

TABLE OF BAUD RATE QUANTUM NUMBERS

The following is a table that gives seventeen of the most commonly used **baud** rates, along with their quantum value (for POKEing) and percent error. Although only seventeen different **baud** rates are shown here, any integer from Ø through 255 may be POKE'd into the proper address (1144+s), and each will give a different **baud** rate.

Average APPLE II Frequency	1.Ø2Ø4842 MHz
Period	.979926 microseconds
Jitter	+139.7 nanoseconds every 65th cycle

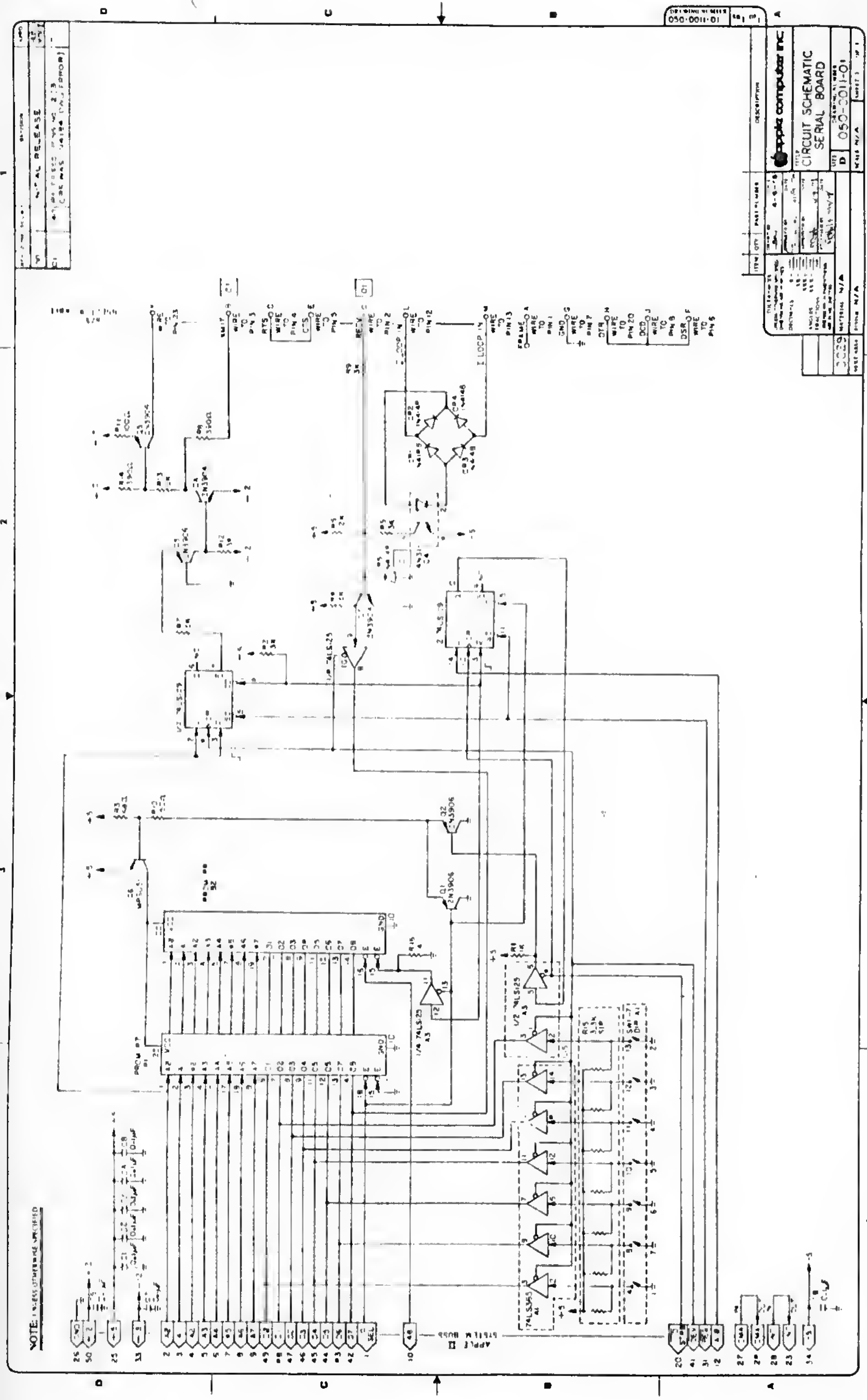
Baud Rate Loop Quantum** 53 APPLE II Cycles (51.94 microseconds)

Baud Rate	Quantum No. (Hex)	Quantum No. (Dec)	Period (microsec.)	Actual Baud Rate	% Error
75	$ ØØ	*** Ø	13296	75.2Ø	+ .28
9Ø	$ D6	214	11115	89.96	− .Ø37
11Ø*	$ BØ	176	9141	1Ø9.4	− .548
134.5*	$ 9Ø	144	7479	133.7	− .586
15Ø	$ 8Ø	128	6648	15Ø.4	+ .28
24Ø	$ 5Ø	8Ø	4155	24Ø.67	+ .28
3ØØ*	$ 4Ø	64	3324	3ØØ.85	+ .28
45Ø	$ 2B	43	2234	447.74	− .51
6ØØ	$ 2Ø	32	1662	6Ø1.7	+ .28
9ØØ	$ 15	21	1Ø91	916.8	+1.86
12ØØ*	$ 1Ø	16	831	12Ø3.4	+ .28
18ØØ	$ ØB	11	571.3	175Ø.2	−2.78
24ØØ*	$ 8	8	415.5	24Ø6.8	+ .28
36ØØ	$ 5	5	259.7	385Ø.59	+6.96
48ØØ*	$ 4	4	2Ø7.7	4813.6	+ .28
96ØØ*	$ 2	2	1Ø3.9	9627.2	+ .28
192ØØ*	$ 1	1	51.9	19254.4	+ .28

*DIP switch selectable

$$\text{**Quantum} = \frac{1}{(.ØØØØ5194) * \textbf{Baud}\text{ Rate}}$$

***The quantum number zero is treated as 256 ($1ØØ).




```
C100           * * * * * * * * * * * * * * * * * * * * * * * * * * *
C100           *                                                   *
C100           *           SERIAL PRINTER / REMOTE INPUT           *
C100           *          INTELIGENT INTERFACE FIRMWARE            *
C100           *                                                   *
C100           *       REVISION 3               JULY 21, 1978      *
C100           *              BY JAMES R. HUSTON                   *
C100           *                (P7-03, P8-01)                     *
C100           * * * * * * * * * * * * * * * * * * * * * * * * * * *
C100           ;
C100           ;
C100           *    ZERO PAGE EQUS    *
C100           ;
C100           ;
C100           CH       EQU  $24         ;CURSOR HORIZONTAL POSITION
C100           BASL     EQU  $28         ;BASE SCREEN AODRESS POINTER
C100           NCOUNT   EQU  $35         ;BIT COUNTER
C100           TEMPX    EQU  $35         ;TEMPORARY BUFFER FOR $CN
C100           CASE     EQU  $35         ;FOR LOWER-UPPER CASE MODES
C100           CSWL     EQU  $36         ;LOW BYTE OF CHAR OUT VECTOR
C100           CSWH     EQU  $37         ; HI BYTE OF CHAR OUT VECTOR
C100           KSWL     EQU  $38         ;LOW BYTE OF CHAR IN VECTOR
C100           KSWH     EQU  $39         ; HI BYTE OF CHAR IN VECTOR
C100           A1L      EQU  $3C         ;MONITOR AODRESS BUFFER
C100           CKSUM    EQU  $42         ;CHECK SUM BUFFER
C100           COUNT    EQU  $43         ;BYTE COUNT BUFFER FOR BATCH
C100           ;
C100           ;
C100           *   GENERAL EQUATES    *
C100           ;
C100           PICK     EQU  $95         ;CONTROL-U
C100           STACK    EQU  $100        ;SYSTEM STACK BLOCK
C100           INBUFF   EQU  $200        ;SYSTEM INPUT BUFFER
C100           KBO      EQU  $C000       ;KEYBOARD INPUT
C100           KBOSTRB  EQU  $C010       ;KEYBOARD CLEAR
C100           OEV      EQU  $C080       ;DEVICE ACCESS
C100           ROMSW    EQU  $CFFF       ;DISABLES CO-RESIOENT $C800 ROMS
C100           ;
C100           ;
C100           *    SLOT VARIABLES    *
C100           ;
C100           BRATE    EQU  $478-$C0    ;THE BAUO QUANTUM NUMBER
C100           STPBITS  EQU  $4F8-$C0    ;THE NUMBER OF STOP BITS
C100           STATUS   EQU  $578-$C0    ;PARITY / CHECKSUM / MOOES
C100           COL      EQU  $5F8-$C0    ;COLUMN POSITION COUNTER
C100           OLOBYTE  EQU  $678        ;BUFFER FOR INPUT (LAST IN)
C100           BYTE     EQU  $678-$C0    ;BUFFER FOR INP/OUT DATA
C100           NO       EQU  $6F8        ;BUFFER FOR OEVICE NUMBER ($NO)
C100           PWOTH    EQU  $6F0-$C0    ;PRINTER (FORMAT) WIOTH
C100           PARITY   EQU  $778        ;BUFFER FOR PARITY CALC.
C100           NBITS    EQU  $778-$C0    ;THE NUMBER OF DATA BITS
C100           MSLOT    EQU  $7F8        ;BUFFER FOR HI SLOT AOOR
C100           FLAGS    EQU  $7F8-$C0    ;FLAGS FOR ALL MODES.
C100           ;
C100           ;
C100           * MONITOR SUBROUTINES *
C100           ;
C100           WAIT     EQU  $FCA8       ;OELAY SUBROUTINE
C100           NXTA1    EQU  $FCBA       ;INCREMENTS A1 ANO COMPARES TO A2
C100           CROUT    EQU  $F08E       ;OUTPUTS A CARRAGE RETURN
C100           COUT1    EQU  $F0F0       ;CHARACTER OUT
C100           PRERR    EQU  $FF20       ;OUTPUTS "ERR"
C100           IORTS    EQU  $FF58       ;KNOWN "RTS" LOCATION
C100                    PAGE
C100           ;
C100           ;
C100 2C 58 FF  OFLTENTR BIT  IORTS       ;SET THE V-FLAG  (OEFAULT ENTRY)
C103 70 04              BVS  ENTRY       ; BRANCH ALWAYS
C105 38        SERIN    SEC              ;SERIAL INPUT ENTRY
C106 90 FE              BCC  *
C108                    ORG  *-I
C107 18        SEROUT   CLC              ;SERIAL OUTPUT ENTRY
C108 B8                 CLV
C109 08        ENTRY    PHP
C10A 78                 SEI
C10B 86 35              STX  TEMPX       ;SAVE INPUT BUFFER INOEX
C10D 48                 PHA              ;SAVE REOISTERS
C10E 8A                 TXA
C10F 48                 PHA
C110 98                 TYA
C111 48                 PHA
C112 AD FF CF           LOA  ROMSW       ;SWITCH OUT ALL CO-RESIOENT $C800 ROMS
C115 20 58 FF           JSR  IORTS       ;OETERMINE SLOT AOORESS
C118 BA                 TSX
C119 BD 00 01           LOA  STACK,X     ;RECOVER HIGH SLOT AOORESS FROM STACK
C11C 8D F8 07           STA  MSLOT
C11F 0A                 ASL  A           ;CREATE $NO FROM SLOT AOOR ($CN)
C120 0A                 ASL  A
C121 0A                 ASL  A
C122 0A                 ASL  A
C123 8D F8 06           STA  NO
C126 A8                 TAY              ;PUT $NO INTO REOISTER-Y ALSO
C127 68                 PLA
C128 68                 PLA
C129 68                 PLA              ;RECOVER CHARACTER
C12A 28                 PLP              ;RECOVER STATUS FLAGS
C12B 9A                 TXS              ;RESTORE STACK POINTER
C12C AE F8 07           LOX  MSLOT       ;PUT THE $CN NUMBER IN REG. X
C12F 48                 PHA              ;SAVE THE CHARACTER
C130 50 03              BVC  NORMIO
C132 20 00 C8  OODEF    JSR  OEFAULT     ;SET ALL SLOT OEPENOANT LOCATIONS
C135 90 43     NORMIO   BCC  SEROUTI     ;BRANCH IF NOT INPUT MOOE
C137 AD 78 06  SERIN1   LOA  OLOBYTE
C13A 48                 PHA              ;SAVE FOR ESC TEST
C13B 08                 PHP
C13C 30 0F              BMI  GETIN       ;IGNORE OLO BYTE IF UPPER CASE
C13E A4 35              LOY  TEMPX       ;GET INPUT BUFFER INOEX VALUE.
C140 F0 0B              BEQ  OETIN       ;OON'T MOOIFY BUFFER IF 1ST CHAR.
C142 88                 OEY              ; OR IF OLOBYTE OOESN'T MATCH
C143 D9 00 02           CMP  INBUFF,Y    ; WHAT'S IN THE INPUT BUFFER.
C146 D0 05              BNE  GETIN
C148 09 E0              ORA  #$E0        ;MAKE IT LOWER CASE PROPER!
C14A 99 00 02           STA  INBUFF,Y
C14D 20 40 C8  GETIN    JSR  SHFTIN      ;OO GET THE SERIAL INPUT.
C150 28                 PLP              ;RESTORE CARRY ANO INTERUPT FLAGS
C151 68                 PLA              ;(OLO BYTE)
C152 49 9B              EOR  #$9B        ;IS IT "ESC"
C154 D0 01              BNE  OOCASE      ;ALL INPUT FOLLOWING "ESC" MUST
C156 18                 CLC              ; UPPER CASE.
C157 A9 0F     OOCASE   LOA  #$0F        ;BIT 5=0 IF UPPER CASE ALFA
C159 90 03              BCC  CAPSONLY    ;BRANCH IF AFTER "ESC"
C15B 1D 38 07           ORA  FLAGS,X     ;IF BIT 5=1 THEN LOWER CASE IS ENABLEO
C15E 85 35     CAPSONLY STA  CASE
```

```
C160  A8                     PLA              ;RECOVER CHAR AT CURSOR
C161  A4 24                  LDY   CH         ;GET CURSOR HORIZONTAL INDEX
C163  08                     PHP              ;SAVE CARRY
C164  4C DC C0               JMP   INFINSH    ;(HO ROOM LEFT IH THIS R.O.M.)
C167                ;
C167                ;
C167  A0 00         NOVID    LDY   #$D
C169  BD 38 05               LDA   COL.X      ;CHECH FOR COLUMH
C16C  FD 38 06               SBC   FWOTH,X    ; WIDTH WITHIN 8 CHARACTERS
C16F  C9 F8                  CMP   #$F8       ; OF PRIHTER WIOTH
C171  90 03                  BCC   SETCH      ;BRAHCH IF NOT
C173  69 27                  ADC   #$27       ;FORM 32-39 FOR BASIC LIST
C175  A8                     TAY              ; FORMATTIND...
C176  38            SETCH    SEC
C177  4C 21 C9               JMP   EXIT2
C17A                         PAGE
C17A                ;
C17A                ;
C17A  BD B8 04      SFROUT1  LOA   STATUS,X
C17D  10 1F                  BPL   OUT1       ;CHECK FOR "AFTER KEYIH" FLAG
C17F  29 7F                  AND   #$7F       ;RESET "AFTER KEYIN" FLAG
C181  9D B8 04               STA   STATUS,X
C184  BD 38 07               LDA   FLAGS,X
C187  4A                     LSR   A          ;CARRY SET OH LIHE FEED MODE
C188  68                     PLA              ;RECOVER CHARACTER
C189  49 8D                  EOR   #$80       ;CHECK FOR CARRADE RETURN
C18B  D0 09                  BHE   DOHE
C18D  9D 38 05               STA   COL,X      ;FIX THE COLUMN COUHT
C190  90 04                  BCC   OOHE       ;BRANCH IF HOT AUTO LF MODE.
C192  A9 8A                  LOA   #$8A       ;LOAD AHD OUTPUT A LIHE FEED
C194  B0 34                  BCS   CTRL       ;BRAHCH ALWAYS
C196  BD 38 07      OONE     LDA   FLAGS,X    ;CHECK VIOEO BIT
C199  30 CC                  BMI   NOVID      ; NO VIDEO EXIT
C19B  4C F9 C8               JMP   VIDEO      ; OUTPUT VIDEO
C19E                ;
C19E                ;
C19E  BD 38 07      OUT1     LDA   FLAGS,X    ;DOH'T MESS WITH THE CURSOR
C1A1  10 14                  BPL   CURSOK     ; IF THE VIDEO IS ON
C1A3  A5 24                  LDA   CH
C1A5  DD 38 05               CMP   COL,X      ;BRAHCH IF CH>COL
C1A8  B0 00                  BCS   CURSOK
C1AA  C9 11                  CMP   #$11
C1AC  B0 09                  BCS   CURSOK
C1AE  09 F0                  ORA   #$FU       ;MODIFY CH MOD(I6) IF APSOFT
C1B0  3D 38 05               ANO   COL,X      ; OR MOD(B) IF INTEGER BASIC
C1B3  65 24                  ADC   CH
C1B5  85 24                  STA   CH
C1B7  BD 38 05      CURSOK   LDA   COL,X
C1BA  C5 24                  CMP   CH         ;TAB CHECK
C1BC  68                     FLA              ;RECOVER CHARACTER
C1BD  B0 03                  BCS   CTRLTST    ;BRANCH IF HO TAB
C1BF  48                     PHA
C1C0  A9 A0                  LDA   #$AD       ;GENERATE SPACES
C1C2  2C 58 FF      CTRLTST  BIT   IORTS
C1C5  F0 03                  BEQ   CTRL       ;DON'T COUNT CTRL CHAR
C1C7  FE 38 05               INC   COL,X      ; IN THE COL COUNT
C1CA  08            CTRL     PHP              ;SAVE P-STATUS
C1CB  48            CRLF     PHA
C1CC  9D B8 05               STA   BYTE,X     ;CHAR TO SHIFT-REG
C1CF  20 AA C9               JSR   SHOUT      ;GO SHIFT IT OUT
C1D2  68                     FLA              ;RECOVER CHARACTER
C1D3  49 8D                  EOR   #$8D       ;CARRAGE RETURH CHECK
C1D5  D0 16                  BHE   STOP       ;BRANCH IF NOT CR
C1D7  9D 38 05               STA   COL,X      ;RESET COLUMN COUNTERS
C1DA  85 24                  STA   CH
C1DC  BD 38 07               LOA   FLAGS,X    ; MUST CHECK FOR DELAY OPTION
C1DF  48                     PHA
C1E0  29 40                  ANO   #$40
C1E2  D0 03                  BHE   LFCHK
C1E4  20 A8 FC      OLAY     JSR   WAIT
C1E7  68            LFCHK    FLA              ;GET FLAGS AGAIH
C1E8  4A                     LSR   A
C1E9  A9 8A                  LDA   #$BA       ;LOAO A LIHE FEED
C1EB  B0 DF                  BCS   CRLF       ;OUTPUT IT IF LIHE-FEED MOOE
C1ED  BD 38 06      STOP     LOA   PWOTH,X    ;CHFCH FOR COL-MAX
C1F0  F0 09                  BEQ   HXTCHR     ;IF PWOTH=0 THEN HEVER MUX IN A CR.
C1F2  FD 38 05               SBC   COL,X
C1F5  B0 04                  BCS   NXTCHR     ;BRAHCH ON COL < PWOTH
C1F7  A9 8D                  LOA   #$80       ;MUX IN A CARRAGE RETURN
C1F9  90 D0                  BCC   CRLF       ;BRANCH ALWAYS
C1FB  28            NXTCHR   FLP
C1FC  90 B9                  BCC   CURSOK     ;BRANCH ON TAB
C1FE  B0 96                  BCS   OONE       ;FIHISH OUTPUT
C200                         PAGE
C200                ;
C200                ;
C200                         ORD   $C800
C800   ********              OFI   SERIAL2
C800                ;
C800                ;
C800                * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C800                ;  THIS IS THE STATUS DEFAULT ROUTINE.  BAUO, WIDTH, CRLF,
C800                ; AHD AUTO CR ARE SET ACCOROIHD TO THE SWITCH SETTINGS:
C800                ;       SWITCH(ES)            FUHCTION
C800                ;         1,2,3                 SELECT 1 OF 8 BAUO RATES.
C800                ;         4                     OELAY AFTER CR OPTION
C800                ;         5,6                   SELECT 1 OF 4 COLUMN WIOTHS.
C800                ;         7                     LINE FEEO AFTER CARRADE RETURN.
C800                ;
C800                ;  ON ENTRY X=$CN, A=$CH. Y MAY BE ANYTHIND.  SWITCH MUST
C800                ; BE SET TO OESIRED COHOITIONS.
C800                ;
C800                ;  ON EXIT X=CN, REGS A&Y ARE MODIFIED. THE CARRY FLAG IS
C800                ; CLEAR IF ENTFREO VIA "HEYIN" OTHERWISE IT IS SET (CAUSING
C800                ; AN OUTPUT TO THE OEVICE). "KEYIN"MAY BE SET AFTER "COUT".
C800                * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C800                ;
C800                ;
C800  B9 80 CD      OEFAULT  LDA   OEV,Y      ;READ SWITCH
C803  48                     PHA              ;SAVE SWITCH STATUS
C804  29 07                  AHD   #$7
C806  A8                     TAY              ;OET D-7 IN Y
C807  B9 31 C9               LOA   BTAB,Y     ;GET THE BAUC QUANTUM
C80A  9D B8 03               STA   BRATE,X    ;SET BAUO RATE
C80D  68                     FLA              ;RECOVER SWITCH STATUS
C80E  0A                     ASL   A
C80F  0A                     ASL   A
C810  2A                     ROL   A          ;PREPARE FLAGS
C811  48                     PHA
C812  90 02                  BCC   MBE40      ;OETERMINE VIOEO BIT
C814  09 80                  ORA   #$BO
C816  29 0F         MBE40    ANO   #$OF       ;SET CAPS-LOCK
C818  9D 38 07               STA   FLAGS,X    ;SAVE FLAGS
```



```
C81B  68                     PLA
C81C  2A                     ROL   A
C81D  2A                     ROL   A          ;DETERMINE WHICH COLUMN WIDTH
C81E  29 03                  AND   #3
C820  A8                     TAY              ;TRANSFER D-3 TO Y
C821  B9 39 C9               LDA   WDTAB,Y    ;GET WIOTH FROM TABLE
C824  9D 38 06               STA   FWDTH,X    ;SAVE COLUMH WIOTH
C827  A9 02                  LDA   #$2        ;SET THE NUMBER OF STPBITS
C829  9D 38 04               STA   STPBITS,X
C82C  A9 09                  LOA   #$9        ;AHD THE OATA BITS
C82E  9D B8 06               STA   NBITS,X
C831  A9 07                  LDA   #$7        ;LAST, THE PARITY STATUS
C833  9D B8 04               STA   STATUS,X
C836  E4 37                  CPX   CSWH       ;SET CHARACTER OUT VECTOR
C838  D0 09                  BHE   NOTOUT     ; OR...
C83A  C5 36                  CMP   CSWL
C83C  F0 05                  BEQ   NOTOUT
C83E  85 36                  STA   CSWL
C840  18            NOTIHP   CLC              ;(INDICATE OUTPUT)
C841  90 09                  BCC   DOODATA
C843  E4 39         NOTOUT   CPX   KSWH       ; SET KEY IN VECTOR.
C845  D0 F9                  PNE   NOTINP
C847  A9 05                  LOA   #>SERIN
C849  85 38                  STA   KSWL
C84B  38            PARERR   SEC              ;(INOICATE INPUT)
C84C  60            GOODATA  RTS
C84D                         PAGE
C84D                ;
C84D                * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C84D                ;  THIS IS THE SERIAL INPUT ROUTIHE.  ON ENTRY X=$CN.
C84D                ; REGISTERS A & Y MAY BE AHYTHINO. BRATE (+$CH) MUST
C84D                ; CONTAIN THE BAUO RATE HUMBER. NBITS (+$CN) MUST CONTAIH
C84D                ; THE HUMBER OF BITS (INCLUOIND THE START BIT), ONE STOP
C84D                ; BIT IS ASSUMED.  $HD MUST BE IN 'ND' (+$CN), BIT 1 OF
C84D                ; FLAGS INDICATES PARITY OPTIOH. (I=HO PARITY, D=PARITY)
C84D                ; IF PARITY IS SELECTED, BIT D OF FLAGS INDICATES 000 OR
C84D                ; EVEN PARITY (D=EVEN PARITY, I=ODD PARITY).
C84D                ;  ON EXIT DATA IS IN BYTE (+$CN) (NOTE: FOR LESB  THAN 8
C84D                ; BITS, THE REMAIHING HIGH ORDER BITS ARE SET TO ONE. I.E.
C84D                ; IF HUMBER OF BITS IS SET TO 6, ONE START BIT AND 5 DATA; .
C84D                ; THE RETURNED DATA WILL BE I11XXXXX BINARY.),
C84D                ; X=$CN, AND Y=D.  IF THE ROUTINE WAS INTERUPTED BY THE
C84D                ; KEYBOARD OR THERE WAS A PARITY ERROR, THE CARRY FLAD IS
C84D                ; SET.  ON ERROR OR KEYSTROKE: X=$CN, A & Y ARE UHDEFINEO.
C84D                * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C84D                ;
C84D  78            SHFTIN   SEI              ;NO HARD INTERRUPTIONS, PLEASE
C84E  BD B8 D4               LDA   STATUS,X   ;PREPARE FOR PARITY/STOP BIT OPTIONS
C851  29 03                  AHD   #$3        ; IF BIT ONE IS ZERO THEN PARITY, IF BIT ZERO IS
C853  4A                     LSR   A          ; ZERO THEN EVEN PARITY
C854  6E 78 D7               ROR   PARITY     ;INITIALIZE PARITY
C857  48                     PHA              ;IF A=O THEN PARITY IS EXPECTED
C858  BD B8 D6               LDA   NBITS,X
C85B  85 35                  STA   NCOUHT     ;INITIALIZE BIT COUNT
C85D  AC FB D6               LDY   ND
C860  38                     SEC              ;SET CARRY FOR BTART BIT
C861  B0 08                  BCS   GFTBIT     ;BRAHCH ALWAYS
C863  AC FB D6      NXTIN    LDY   HO
C866  C6 35                  DEC   NCOUHT     ;ARE ALL THE BITS IN YET?
C868  F0 4D                  BEQ   CHHPAR     ;YES, GET PARITY/STOP BIT...
C86A  18                     CLC              ;HO, MORE TO COME
C86B  B9 8D CD      GETBIT   LDA   DEV,Y
C86E  9D 1F                  BCC   SAVBIT     ;BRAHCH IF DATA
C870  1D DD                  BPL   GOTSTRT    ;BRAHCH IF START BIT
C872  AD DD CD               LDA   HBO        ;MAYBE THE KEYBOARD WANTS ATTENTIOH
C875  1D F4                  BPL   GETBIT     ;NO, WAIT FOR THE START BIT
C877  9D B8 05               STA   BYTE,X
C87A  2C 10 C0               BIT   HBDSTRB    ;CLEAR KEYBOARD
C87D  68                     PLA              ;YES, BETTER FIX THE STACK BEFORE RETURN
C87E  60                     RTS              ;  THE HEYBOARO INTERRUPTED THE INPUT!
C87F                ;
C87F  BD B8 03      GOTSTRT  LDA   BRATE,X
C882  A0 09                  LDY   #$9        ;MUST CREATE A DELAY SUCH THAT OATA IS TAKEH
C884  88            SWAIT    DEY              ;  AT THE CEHTER OF THE SIGHAL.(BRANCH ALWAYS)
C885  D0 FD                  BHE   SWAIT
C887  E9 01                  SBC   #$1        ;OOHE WAITIHD AROUNO?
C889  F0 08                  BEQ   NXTIH      ;YES, IT'S GET THAT OATA TIME
C88B  A0 0E                  LOY   #$E        ;NO, WAIT AHOTHER 78 CYCLE TIMES
C88D  D0 F5                  BHE   SWAIT      ;BRAHCH ALWAYS
C88F                ;
C88F  A8            SAVBIT   TAY
C890  2A                     ROL   A          ;SHIFT THE DATA INTO CARRY
C891  7E B8 D5               ROR   BYTE,X     ;THEN PUT IT AWAY.
C894  98                     TYA
C895  4D 78 07               EOR   PARITY     ;UPDATE THE PARITY
C898  8D 78 07               STA   PARITY
C89B  BD B8 D3               LOA   PRATE,X    ;GET THE BAUO DELAY NUMBER
C89E  38                     SEC
C89F  E9 01         WAIT1    SBC   #$1        ;TIME OUT THE BAUO RATE
C8A1  F0 C0                  BEQ   HXTIN      ;BRAHCH IF DONE
C8A3  A0 D9                  LOY   #$9
C8A5  88            WAIT53   OEY              ;OELAY=53*(BRATE-1) CYCLES
C8A6  D0 F0                  BHE   WAIT53
C8A8  F0 F5                  BEQ   WAIT1
C8AA                ;
C8AA  68            CHKPAR   PLA              ;PARITY OR HO PARITY, THAT IS THE QUESTION
C8AB  D0 0R                  BHE   MOO        ;BRAHCH IF HO PARITY
C8AD  B9 80 C0               LDA   DEV,Y      ;GET THE PARITY BIT
C8B0  4D 78 D7               EOR   PARITY     ;DOES THE ONE SENT MATCH THE CALCULATED?
C8B3  30 96                  BMI   PARERR     ;HOT IF THE MIHUS FLAG IS ON
C8B5  B9 80 CU      MOD      LOA   DEV,Y      ;MAKE SURE THE SIDNAL IS HIDH BEFORE
C8B8  10 FB                  BPL   MOO        ;  RETURNINO...
C8BA  A9 0A                  LOA   #$A
C8BC  FD B8 D6               SBC   NBITS,X    ;HOW MANY BITS TO MAKE A WHOLE BYTE?
C8BF  A8                     TAY              ;PUT THE LEFTOVER BIT COUNT IN REGISTER-Y
C8C0  88            MOD1     OEY              ;SHIFT IN A ONE?
C8C1  F0 89                  BEQ   GOODATA    ;NO, SEND THE OATA BACK TO CALLER
C8C3  7E B8 D5               ROR   BYTE,X     ;YES, MAKE THE HIGH ORDER BIT A ONE.
C8C6  38                     SEC
C8C7  B0 F7                  BCS   MOO1       ;BRANCH ALWAYS
C8C9                ;
C8C9                ;
C8C9   ********              OFI   SERIAL3
C8C9                ;
C8C9                ;
C8C9                *   THE FOLLOWING ARE VARIOUS EXIT ROUTINES.  *
C8C9                ;
C8C9                ;
C8C9  68            VIOEO    PLA
C8CA  A8                     TAY
C8CB  68                     PLA              ;RESTORE ALL REGISTERS
C8CC  AA                     TAX
C8CD  68                     PLA
```

```
CBCF 09 80              ORA   #$80
CBD0 49 E0              EOR   #$E0
CBD2 C9 1F              CMP   #$1F
CBD4 90 02              BCC   MXTOUT
CBD6 49 E0              EOR   #$E0
CBD8 28        MXTOUT   PLP
CBD9 4C F0 FD           JMP   COUT1
CBDC                    ;
CBDC 91 28     INFINSH  STA   (BASL),Y     ;REPLACE FLASHING CURSOR
CBDE BD B8 05           LOA   BYTE,X       ;GET MOST RECENT INPUT
CBE1 09 80              ORA   #$80
CBE3 C9 95              CMP   #PICK        ;SCREEN PICK (CONTROL-U)?
CBE5 D0 02              BNE   NOTPICK
CBE7 B1 28              LOA   (BASL),Y     ;GET SCREEN CHARACTER
CBE9 C9 E0     NOTPICK  CMP   #$E0         ;OO NOTNING TO UPPER AN NON-ALEA
CBEB 90 11              BCC   EXIT1
CBED 25 35              ANO   CASE         ;MAKE LOWER TO UPPER IF LC NOT ENABLEO
CBEF C9 E0              CMP   #$E0         ;RETURN LOWER CASE IF ENABLEO
CBF1 90 0B              BCC   EXIT1
CBF3 A4 37              LOY   CSWH         ;IF VIOEO OUT ONLY, TNEN CHAR
CBF5 C0 F0              CPY   #$F0         ; SHOULO BE OISPLAYEO INVERSE.
CBF7 F0 03              BEQ   IVERSE
CBF9 29 7F              AND   #$7F         ; OTNERWISE JUST STRIP BIT 7
CBFB AC 58 FF           LOY   IORTS        ;(OUMMY LOY ABSOLUTE)
CBFE                    ORG   *-2
CBFE 29 1F     IVERSE   ANO   #$1F         ;MAKE IT INVERSE VIOEO
CC00 8D 78 06  EXIT1    STA   OLOBYTE      ;SAVE FOR NEXT INPUT TO CORRECT BUFFER
C901 28                 PLP                ;IF CARRY CLR, OO "ESC" FUNCTION
C902 B0 15              BCS   NOTESC
C904 C9 CC              CMP   #$CC         ;IS IT AN ASCII "L"
C906 30 09              BMI   UPPER        ;CAPSLOCK COMMANO IF >L
C908 D0 0D              BNE   NOTESC
C90A A9 20              LOA   #$20         ;ENABLE LOWER CASE
C90C 1D 38 07           ORA   FLAGS,X
C90F D0 05              BNE   STORFLGS     ;BRANCN ALWAYS TAKEN
C911 A9 0F     UPPER    LOA   #$0F         ;OISABLE LOWER CASE
C913 3D 38 07           ANO   FLAGS,X
C916 9D 38 07  STORFLGS STA   FLAGS,X
C919 18        NOTESC   CLC                ;INOICATE INPUT RETURN
C91A BD B8 04           LOA   STATUS,X     ;INOICATE AFTER INPUT (TO OUTPUT)
C91D 09 80              ORA   #$80
C91F 9D B8 04           STA   STATUS,X
C922 AC 58 FF           LOY   IORTS        ;(ANOTHER OUMMY)
C925                    ORG   *-2
C923 84 24     EXIT2    STY   CH           ;RESTORE CURSOR HORIZONTAL
C925 68                 PLA
C926 A8                 TAY
C927 68                 PLA                ;RESTORE REOISTERS
C928 AA                 TAX
C929 68                 PLA                ; EXCEPT REO. A IF INPUT RETURN
C92A B0 03              BCS   RETOUT
C92C AD 78 06           LOA   OLOBYTE      ;RETURN NEW INPUT
C92F 28        RETOUT   PLP
C930 60                 RTS                ;RETURN TO CALLER
C931                    PAGE
C931                    ;
C931                    ;
C931                    *     BAUO  TABLE      *
C931                    ;
C931 80        BTAB     OFB   $80          ;  110    BAUO
C932 90                 OFB   $90          ;  134.5  BAUO
C933 40                 OFB   $40          ;  300    BAUO
C934 10                 OFB   $10          ;  1200   BAUO
C935 08                 OFB   $8           ;  2400   BAUO
C936 04                 OFB   $4           ;  4800   BAUO
C937 02                 OFB   $2           ;  9600   BAUO
C938 01                 OFB   $1           ;19200    BAUO
C939                    ;
C939                    *     WIOTH  TABLE     *
C939 29        WOTAB    OFB   $29          ;   40    COLUMNS
C93A 48                 OFB   $48          ;   72    COLUMNS
C93B 50                 OFB   $50          ;   80    COLUMNS
C93C 84                 OFB   $84          ;  132    COLUMNS
C93D                    PAGE
C93D                    ;
C93D                    ;
C93D                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C93D                    ;  THESE ARE THE BATCH ROUTINES.  ON EITNER ENTRY (IN OR
C93D                    ; OUT), LOCATIONS $3C & $3D MUST CONTAIN THE BEGINNING RAM
C93D                    ; AOORESS, AND LOCATIONS $3E & $3F MUST CONTAIN THE ENO-
C93D                    ; RAM AOORESS FOR THE BATCH MOVE OPERATION.  THE CARO MUST
C93D                    ; HAVE BEEN INITIALIZEO ANO PARITY & CHECK SUM OPTIONS SET
C93D                    ; PRIOR TO ENTRY.  ON ENTRY A,X,&Y REGISTERS MAY CONTAIN
C93D                    ; ANYTHING.
C93D                    ;  ON EXIT REGISTERS ARE UNOEFINEO.
C93D                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C93D                    ;
C93D                    ;
C93D 2C 58 FF  INPUT    BIT   IORTS        ;SET VFLAO TO INOICATE INPUT
C940 90 FF              BCC   *
C942                    ORG   *-1
C941 B8        OUTPUT   CLV                ;GET READY FOR BATCH MOVE
C942 AE F8 07           LOX   MSLOT
C945 BD B8 04           LOA   STATUS,X
C948 4A                 LSR   A
C949 4A                 LSR   A
C94A 4A                 LSR   A            ;OETERMINE CHECKSUM
C94B 08        BATCH    PHP
C94C A0 00              LOY   #$0
C94E 84 42              STY   CKSUM        ;INITIALIZE CHECKSUM ANO COUNT
C950 84 43              STY   COUNT
C952 70 2D              BVS   MOVIN
C954 B1 3C     MOVOUT   LOA   (A1L),Y      ;GET THE DATA TO BE OUTPUT
C956 9D B8 05           STA   BYTE,X
C959 45 42              EOR   CKSUM        ;UPOATE CHECKSUM
C95B 85 42              STA   CKSUM
C95D 20 AA C9           JSR   SHOUT        ;GO OUTPUT IT
C960 20 BA FC           JSR   NXTA1        ;INCREMENT POINTERS
C963 B0 11              BCS   FINISH
C965 E6 43              INC   COUNT        ;UPOATE BYTE COUNT
C967 D0 EB              BNE   MOVOUT
C969 28                 PLP
C96A B0 DF              BCS   BATCH        ;OO CHECKSUM IF CARRY CLR
C96C A5 42              LOA   CKSUM        ;GET THE CHECKSUM FOR THE LAST
C96E 9D B8 05           STA   BYTE,X       ; BLOCK OUTPUT
C971 20 AA C9           JSR   SHOUT        ;OUTPUT THE CHECKSUM
C974 90 D5              BCC   BATCH        ;BRANCH ALWAYS
C976 68        FINISH   PLA
C977 B0 07              BCS   RETBATCH
C979 20 8E FD           JSR   CROUT
C97C 20 20 FF           JSR   PRERR        ;GO PRINT "ERR"
C97F 38                 SEC
C980 60        RETBATCH RTS
```



```
C981                    ;
C981                    ;
C981 20 40 CB  MOVIN    JSR   SHFTIN       ;GET THE INPUT
C984 30 21              BMI   ERR1
C986 BD B8 05           LOA   BYTE,Y
C989 91 3C              STA   (A1L),Y      ;PUT THE INPUT AWAY
C98B 45 42              EOR   CKSUM
C98D 85 42              STA   CKSUM        ;UPOATE THE CHECKSUM
C98F 20 BA FC           JSR   NXTA1        ;INCREMENT RAM POINTER
C992 B0 E2              BCS   FINISH
C994 E6 43              INC   COUNT        ;UPOATE THE BYTE COUNT
C996 D0 E9              BNE   MOVIN        ;BRANCH IF BLOCK NOT FILLEO
C998 28                 PLP
C999 B0 B0              BCS   BATCH        ;BRANCH IF NO CHECKSUM
C99B 20 40 CB           JSR   SHFTIN       ;GET THE CHECKSUM INPUT
C99E BD B8 05           LOA   BYTE,X
C9A1 C5 42              CMP   CKSUM
C9A3 18                 CLC                ;INOICATE CHECKSUM MOOE
C9A4 F0 A5              BEQ   BATCH
C9A6 48                 PHA
C9A7 18        ERR1     CLC                ;INOICATE ERROR CONOITION
C9A8 90 CC              BCC   FINISH       ;BRANCH ALWAYS TAKEN
C9AA                    PAGE
C9AA                    ;
C9AA                    ;
C9AA                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C9AA                    ;  THIS IS THE SERIAL OUTPUT ROUTINE.    ON ENTRY X=$CN
C9AA                    ; (N IS THE SLOT NUMBER), REGS A AND Y CAN BE ANYTHING.
C9AA                    ; THE OATA TO BE OUTPUT MUST BE IN BYTE (+$CN). IF OATA IS
C9AA                    ; LESS THAN 8 BITS (NBITS<9), THE LEFTOVER HIGH BITS MUST
C9AA                    ; BE SET TO ONES.  (I.E. FOR 7 BITS:  BYTE= 1XXXXXXX)
C9AA                    ; THE NUMBER OF BITS (INCLUOING THE START BIT) MUST BE IN
C9AA                    ; NBITS (+$CN), THE BAUO RATE (OELAY NUMBER) MUST BE IN
C9AA                    ; BRATE (+$CN), & NUMBER OF STOP BITS (+ PARITY BIT) IN
C9AA                    ; STPBITS (+$CN).   $NO MUST BE IN 'NO' (+$CN).
C9AA                    ;  ON EXIT A=0, X=$CN, AND Y=0.  P-STATUS IS MOOIFIEO
C9AA                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C9AA                    ;
C9AA 78        SHOUT    SEI                ;SET FOR NO HARO INTERRUPTIONS
C9AB 18                 CLC
C9AC 08                 PHP
C9AD 38                 SEC
C9AE 08                 PHP
C9AF BC B8 06           LOY   NBITS,X      ;GET THE NUMBER OF BITS/WORO
C9B2 84 35              STY   NCOUNT       ;STORE IT IN THE BIT COUNTER
C9B4 A9 01              LOA   #$1
C9B6 3D B8 04           ANO   STATUS,X     ;FOR EVEN OR OOO PARITY
C9B9 0D F8 06           ORA   NO
C9BC 8D 78 07           STA   PARITY       ;INITIALIZE PARITY
C9BF A9 00              LOA   #$0
C9C1 1E B8 05           ASL   BYTE,X       ;SHIFT FOR START BIT
C9C4 7E B8 05  BITOUT   ROR   BYTE,X
C9C7 2A                 ROL   A            ;BIT 0 OF BYTE -> BIT 0 OF ACCUMULATOR
C9C8 0D F8 06           ORA   NO
C9CB A8                 TAY                ;COPY INTO REGISTER Y
C9CC 29 01              ANO   #$1          ;STRIP ALL BUT THE OATA
C9CE 4D 78 07           EOR   PARITY       ;UPOATE THE PARITY
C9D1 8D 78 07           STA   PARITY       ;ANO SAVE IT AGAIN
C9D4 38        PAROUT   SEC                ;THIS ENTRY POINT FOR STOPBITS AND PARITY
C9D5 B9 81 C0           LOA   DEV+1,Y      ;OATA IS SENT VIA BIT 0 OF THE ADDRESS BUS
C9D8 BD B8 03           LOA   BRATE,X      ;GET THE BAUO NUMBER
C9DB E9 01     BOLAY    SBC   #$1
C9DD F0 07              BEQ   NXTOUT       ;DELAY = 53 * (BAUD NUMBER-1) CYCLE TIMES
C9DF A0 09              LOY   #$9
C9E1 88        BOLY1    OEY
C9E2 D0 FD              BNE   BOLY1
C9E4 F0 F5              BEQ   BOLAY        ;BRANCH ALWAYS TAKEN
C9E6 BC 38 04  NXTOUT   LOY   STPBITS,X    ;TIMING OURING OATA TRANSFER
C9E9 C6 35              OEC   NCOUNT       ;MORE BITS TO OO?
C9EB D0 D7              BNE   BITOUT       ;YES, OO ANOTHER ONE
C9ED 28                 PLP                ;NO, BUT WHAT ABOUT STOP BITS?
C9EE 90 0E              BCC   RETSUB       ;CARRY CLEAR, THEN OONE
C9F0                    ;
C9F0                    ;  STOP BITS (ANO PARITY IF SELECTEO) YET TO OO
C9F0                    ;
C9F0 84 35              STY   NCOUNT       ;PUT THE NUMBER OF STOPBITS IN COUNTER
C9F2 BD B8 04           LOA   STATUS,X     ;NOW TO SEE IF A PARITY BIT IS TO BE SENT
C9F5 29 02              ANO   #$2          ;IF ZERO THEN PARITY IS SENT, ELSE JUST
C9F7 4A                 LSR   A
C9F8 0D 78 07           ORA   PARITY       ;  STOP BITS ARE SENT.
C9FB A8                 TAY                ;NOW Y=$NO+(STOP BIT OR PARITY BIT)
C9FC D0 D6              BNE   PAROUT       ;GO PUT IT OUT!
C9FE A8        RETSUB   TAY                ;THEN SET THE ZERO FLAG
C9FF 60                 RTS
CA00                    ;
```

LABELS

```
003C  A1L        0028  BASL       C94B  BATCH      C908  BOLAY
C9EI  BOLY1      C9C4  BITOUT     03B8  BRATE      C931  BTAB
05B8  BYTE       C15E  CAPSONLY   0035  CASE       0024  CH
CBAA  CHKPAR     0042  CKSUM      0538  COL        0043  COUNT
F0F0  COUT1      C1CB  CRLF       FD8E  CROUT      0037  CSWH
0036  CSWL       C1CA  CTRL       C1C2  CTRLTST    C1B7  CURSOK
C800  OEFAULT    C080  OEV        C100  OFLTENTR   C1E4  OLAY
C157  OOCASE     C132  OOOEF      C196  DONE       C109  ENTRY
C9A7  ERR1       CBFE  EXIT1      C923  EXIT2      C976  FINISH
0738  FLAGS      CB6B  GETBIT     C140  GET1N      CB4C  GOODATA
CB7F  GOTSTRT    0200  INBUFF     C80C  INFINSH    C930  INPUT
FF58  IORTS      C8FC  IVERSE     C000  KBD        C010  KBOSTRB
0039  KSWH       0038  KSWL       C1E7  LFCHK      C816  MBE40
CBB5  MOD        CBC0  MOD1       C9BI  MOVIN      C954  MOVOUT
07F8  MSLOT      C808  MXTOUT     06FB  NO         06BB  NBITS
0035  NCOUNT     C135  NORMIO     C919  NOTESC     CB40  NOTINP
C843  NDTOUT     CBE9  NOTPICK    C167  NOVIO      FCBA  NXTA1
C1FB  NXTCHR     CB63  NXTIN      C9E6  NXTOUT     0678  OLOBYTE
C19E  OUT1       C941  OUTPUT     C84B  PARERR     0778  PARITY
C904  PAROUT     0095  PICK       FF20  PRERR      0638  PWOTH
C980  RETBATCH   C92F  RETOUT     C9FE  RETSUB     CFEF  ROMSW
CB8F  SAVBIT     C105  SERIN      C137  SERIN1     C107  SEROUT
C17A  SEROUT1    C176  SETCH      CB40  SHFTIN     C9AA  SHOUT
0100  STACK      04B8  STATUS     C1E0  STOP       C916  STORFLGS
0438  STPBITS    CB84  SWAIT      0035  TEMPX      C911  UPPER
CBC9  VIOEO      FCA8  WAIT       CB9F  WAIT1      CBA5  WAIT53
C939  WOTAB
```